INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2007; **53**:507–529
Published online 24 July 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1296

A sliding interface method for unsteady unstructured flow simulations

Eric L. Blades[1,*,†] and David L. Marcum[2,‡]

[1]*Computational Simulation and Design Center, Mississippi State University, U.S.A.*
[2]*Department of Mechanical Engineering, Mississippi State University, U.S.A.*

SUMMARY

The objective of this work is to develop a sliding interface method for simulations involving relative grid motion that is fast and efficient and involves no grid deformation, remeshing, or hole cutting. The method is implemented into a parallel, node-centred finite volume, unstructured viscous flow solver. The rotational motion is accomplished by rigidly rotating the subdomain representing the moving component. At the subdomain interface boundary, the faces along the interface are extruded into the adjacent subdomain to create new volume elements forming a one-cell overlap. These new volume elements are used to compute a flux across the subdomain interface. An interface flux is computed independently for each subdomain. The values of the solution variables and other quantities for the nodes created by the extrusion process are determined by linear interpolation. The extrusion is done so that the interpolation will maintain information as localized as possible. The grid on the interface surface is arbitrary. The boundary between the two subdomains is completely independent from one another; meaning that they do not have to connect in a one-to-one manner and no symmetry or pattern restrictions are placed on the grid. A variety of numerical simulations were performed on model problems and large-scale applications to examine conservation of the interface flux. Overall solution errors were found to be comparable to that for fully connected and fully conservative simulations. Excellent agreement is obtained with theoretical results and results from other solution methodologies. Copyright © 2006 John Wiley & Sons, Ltd.

Received 2 August 2005; Revised 30 December 2005; Accepted 28 April 2006

KEY WORDS: relative grid motion; CFD; unstructured grids

1. INTRODUCTION

Time accurate prediction of flow fields about geometries in relative motion is of great interest. Practical applications include turbomachinery, helicopters, tiltrotors, and ship propellers. Liu and

*Correspondence to: Eric L. Blades, Computational Simulation and Design Center, P.O. Box 9627, Mississippi State, MS 39762, U.S.A.
†E-mail: blades@hpc.msstate.edu
‡E-mail: marcum@hpc.msstate.edu

Hill [1] conducted an investigation comparing three approaches for simulating the unsteady flow of a centrifugal compressor, including the Frozen Rotor model, Circumferential Average model, and a sliding mesh model. They concluded that although the sliding mesh model is computationally more intensive, it is the necessary approach to predict the inherently unsteady flow field because only the sliding mesh model was capable of simulating the aerodynamic interaction due to the impeller rotation relative to either the upstream guide inlet vanes or downstream discharge vanes.

Three main approaches have been devised to treat moving bodies or grids with relative motion for unstructured grids. One approach is to remesh the domain, either locally or the entire domain [2]. Even if confined to a local portion of the domain, the remeshing procedure can become computationally expensive, particularly for unsteady simulations that may take several revolutions to establish periodicity. Another is to deform the grid in response to the relative motion of the body [3, 4]. Depending upon the motion, the grid movement may eventually deteriorate the grid quality and remeshing all or part of the domain becomes necessary. Both remeshing and grid deformation allow fully general motion of the components. However, for a third approach, a large class of problems, like those considered herein, the direction of relative motion of the components is known *a priori*. Most rotating machinery simulations, including axial and centrifugal turbomachinery, mixing tanks, ship and aircraft propellers, etc. fall into this class of problems. Thus for the third approach, the remeshing can be avoided and the grid motion can be accomplished by decomposing the domain into subdomains which move relative to one another along judiciously chosen boundaries or interfaces. In this last approach, the problem then becomes how to couple the two subdomains across the chosen boundary or interface.

2. INTERFACE METHODS

There has been much attention devoted to this coupling of subdomains at an interface, even for problems that do not involve relative grid motion. In the Chimera or structured composite overset approach [5], the discretization is comprised of a system of component grids and background grids. The grid components are not required to align with neighbouring components in any special way, can overlap one another, and there is generally a multiple cell overlap. Valid Chimera holes must be cut in each grid within regions that overlap with other grid components, solid bodies, or any other non-flow regions. The Chimera holes serve to identify active and non-active parts of the overall grid. Interpolation stencils have to be created for all inter-grid boundary points. Identification of the interpolation stencil involves a search for donor cells for all points that lie along the Chimera hole boundary. At the inter-grid boundaries, a non-conservative interpolation is typically used to transfer information across the interface. Meakin [6] showed that the formal order of accuracy is maintained with a non-conservative Chimera approach using simple interpolation. Overset grid topologies have been successfully applied to geometrically complex configurations involving moving bodies and relative grid motion [7, 8]. For these types of problems, the hole cutting and donor cell identification must be done each time the grid changes and can be computationally expensive. Research has recently begun to develop a conservative interface for Chimera grids [9, 10]. Wang *et al.* [9] show that on a sufficiently fine mesh, a conservative Chimera and a non-conservative Chimera scheme using simple interpolation converge to the same solution.

Nakahasi *et al.* [11] and Löhner *et al.* [12] have extended the structured overset or Chimera approach to unstructured topologies to treat problems with moving bodies and relative grid motion. The same hole cutting and donor cell identification procedures as previously discussed are required, but implemented on an unstructured topology. Initially the approach was implemented for inviscid simulations and later extended to viscous simulations [13, 14]. Similar to the structured overset approach, the unstructured approach is not conservative across the inter-grid boundaries. Zhang *et al.* [15] have proposed a combination of the previous two approaches: an unstructured overset method that combines grid movement and remeshing. Once the holes have been cut in the overlapping regions, instead of the inter-grid interpolation, the grid is locally remeshed in the hole regions to maintain conservation. However, due to changes in the grid from one time step to the next, the method is not strictly conservative as will be discussed further in the UVI method. In addition to the expense of identifying the holes, the method also incurs the additional expense of remeshing.

Compared to sliding grid techniques for structured grid topologies, the sliding interfaces for unstructured grids are more complex because of the explicit data structure required for connectivity. Even though the grids do not have to align along the subdomain interface boundaries for structured grids, the interpolation is relatively easy to implement due to the implicit connectivity data structure. Unstructured grids, however, result in interfaces that are also unstructured and therefore the interface communication procedures are more complex. Pan *et al.* [16] developed a sliding grid approach suitable for inviscid computations with global flux conservation across subdomain interfaces. The approach is similar to the structured patch grid approach, but for use in an unstructured solver. The drawback to this method is that certain symmetry restrictions are placed on the grid interface in order to maintain global conservation. Yu *et al.* [17] used a sliding interface on a hybrid unstructured grid for a rotor-stator analysis that imposes even further restrictions; requiring that the subdomain grids exactly align at the interface at all time steps. Since the grids at the interface match identically, no interpolation is required. However, the spacing at the interface is dictated by the time step and a change in the time step requires remeshing of the interface region. Both of these approaches defeat one of the main advantages of an unstructured approach, namely the flexibility to handle complex geometries.

Mathur [18] has developed a more general sliding interface for unstructured grids that imposes no restriction on the node placement of the subdomain boundary interfaces and is also conservative. The method takes advantage of the cell-centred scheme for which it is implemented and requires no interpolation across the interface. The overlapping faces at the interface surfaces are replaced by a new set of faces formed by their intersections such that each new face has a unique cell neighbour on the opposite side. However, the method is restricted to sliding boundaries that are cylindrical or conical surfaces of revolution.

An unstructured method for node-centered schemes was developed by Sreenivas *et al.* [19] for tilt-rotor simulations. This method, referred to as the UVI method, employs local grid reconnection to enable relative grid motion. For example, to simulate a rotating propeller, a surface or interface is created to divide the domain into two subdomains: an inner domain representing the propeller and an outer domain representing the rest of the domain of interest. To simulate the motion of the rotating propeller, the cells attached to either the inside or the outside of the interface are deleted, thus leaving a void in the domain. Next, the inner subdomain representing the propeller is rotated as a rigid unit into the desired orientation. Then a local reconnection process [20] is performed to re-generate the deleted cells in order to merge the two subdomains, resulting in one continuous domain. The domain is continuous in the sense that the two subdomains are merged together sharing a common set of nodes at the interface boundary. The UVI method, in principle, is an unstructured implementation of the localized grid distortion and clicking method introduced by Janus [21]. The local-reconnection process reconnects the distorted grid lines at the interface, and the inner grid is essentially clicked into place.

Despite the fact the domain is continuous at the interface boundary, the approach is not strictly conservative. Venkatakrishnan and Mavriplis [22] point out that after the local-reconnection process, the solution, which is stored at the node locations, needs to be modified to satisfy the conservation in time requirement since the control volumes for the nodes or points involved in the reconnection may have changed discontinuously from the previous time level. They proposed a conservative, linearity-preserving interpolation procedure to modify the solution to account for the change in connectivity as well as other possible approaches to update the solution. However, for the UVI method as currently implemented in this work, no adjustments are made to the solution after the reconnection process to account for the change in connectivity.

A limitation with the current implementation of the UVI method is that the grid reconnection can only be done in isotropic regions of the grid, and cannot include any physical boundaries. This may require that the geometry be cut and a gap created to allow the UVI surface to pass through the boundary surface. This gap creates a void in the physical boundary of the surface that is now part of the flow field. Thus, during the volume grid generation process, a volume grid will be generated within the gap. If the gap required by the UVI method is aligned with the free stream, the flow through this gap can pose numerical problems, particularly in supersonic flows. This reconnection limitation is an implementation issue of the reconnection process that can be resolved.

Another limitation for the UVI method is that the local reconnection grid generation process is currently a serial process and therefore utilizes only a single CPU. The overall unstructured grid generation process could be implemented to work in parallel. However, the local-reconnection of a thin layer about a UVI domain is not highly parallelizable. In a parallel computing environment, this creates a bottleneck by forcing all the remaining processors to remain idle and wait on a single processor to perform the local reconnection. This can significantly increase the run-time for simulations involving relative grid motion. In addition, since it is a serial process running on a single CPU, there are memory restrictions that limit the resolution of the UVI surface. To minimize the modification to the geometry, it is obviously desired to make the gap as small as possible. For the reconnection process to work best, the elements inside the gap should be nearly isotropic, which requires the grid spacing in the region of the gap to be approximately half the width of the gap. Thus, additional resolution is wasted on a part of the numerical domain that is not even present in the physical domain.

3. SOLUTION ALGORITHM

Before discussion of the sliding interface method, an overview of the unstructured flow solver into which it is implemented is presented. Both the UVI and sliding interface methods were implemented in the MSU U^{2}NCLE unstructured flow solver [23]. U^{2}NCLE is a parallel flow simulation code that solves the unsteady Reynolds-averaged Navier–Stokes equations for complex geometries represented by multi-element unstructured grids. U^{2}NCLE can solve inviscid, laminar, and high Reynolds number flows for either steady or unsteady conditions in compressible or incompressible flows.

The flow solver, *u*nstructured *un*steady *c*omputation of fie*l*d *e*quations (U^{2}NCLE), is a node-centered, finite volume, implicit scheme applied to general unstructured grids with non-simplical elements. The flow variables are stored at the vertices and surface integrals are evaluated on the median dual surrounding each of these vertices. The non-overlapping control volumes formed by the median dual completely cover the domain, and form a mesh that is dual to the elemental grid. Thus, a one-to-one mapping exists between the edges of the original grid and the faces of the control volumes.

The solution algorithm consists of the following basic steps: reconstruction of the solution states at the control volume faces, evaluation of the flux integrals for each control volume, and the evolution of the solution in each control volume in time.

For the numerical solutions conducted herein, the inviscid fluxes at each face of the control volume are evaluated using the flux-difference splitting technique of Roe [24]. The algebraic flux vector is replaced by a numerical flux function which depends on the reconstructed data on each side of the control volume face:

$$\Phi = \tfrac{1}{2}(F(Q_L) + F(Q_R)) - \tfrac{1}{2}\tilde{A}(Q_R - Q_L)$$

where Q is the dependent variable vector for the conservative variables $Q = [\rho \;\; \rho u \;\; \rho v \;\; \rho w \;\; E]^{\mathrm{T}}$ and $\tilde{A} = \tilde{R}\tilde{\Lambda}\tilde{R}^{-1}$. The matrix $\tilde{R}$ is a matrix constructed from the right eigenvectors of the flux Jacobian matrix, $\partial G/\partial Q$, and $\tilde{\Lambda}$ is a diagonal matrix whose entries contain the absolute values of the flux Jacobian matrix. The $(\tilde{\;})$ quantities are constructed with variables using the Roe averaging procedure. A higher order spatial method is constructed by a 2nd-order reconstruction (extrapolation) of the data on either side of the control volume face

$$\mathbf{q}_f = \mathbf{q}_0 + \phi\nabla\mathbf{q}_0 \cdot \mathbf{r}$$

where $\mathbf{q}_f$ is the reconstructed function, $\nabla\mathbf{q}_0$ is the gradient of primitive variables at the vertex, $\mathbf{r}$ is the vector from the vertex to the midpoint of the edge, and ϕ is a slope-limiting function. The Barth–Jesperson limiter is applied to these terms to prevent oscillations and overshoots in the numerical solution [25]. The one-equation turbulence model of Spalart and Allmaras was used for simulation of turbulent effects in high Reynolds number flows [26].

The temporal discretization is done using a backward Euler implicit scheme

$$\frac{(1+\varphi)V^{n+1}\Delta\overline{Q}^n - \varphi V^{n-1}\Delta\overline{Q}^{n-1}}{\Delta t} + \overline{Q}^n \left[\frac{\Delta V^n - \dfrac{\varphi}{1+\varphi}\Delta V^{n-1}}{\Delta t} \right] + R^{n+1} = 0$$

where V is the volume of the control volume, $\overline{Q}$ is the integral average over the control volume, Δt is the time step, and φ is a parameter that controls the temporal order of accuracy. For a second-order scheme $\varphi = \frac{1}{2}$ and for a first-order scheme $\varphi = 1$. The divergence term in brackets in the preceding equation is a temporal discretization of the change in the volume of the control volume and represents the geometric conservation law (GCL). A Newton iterative time evolution scheme is employed to advance the unsteady solution at each time step which linearizes the equations about a given time level and results in a linear system of equations that must be solved [27]

$$\frac{\partial L(\overline{Q}^{n+1,m})}{\partial \overline{Q}}\Delta\overline{Q}^{n+1,m} = -L(\overline{Q}^{n+1,m})$$

where m is the Newton iteration index and $\Delta Q^{n+1,m} = Q^{n+1,m+1} - Q^{n+1,m}$. L is a vector function given by

$$L(\overline{Q}^{n+1}) = \frac{(1+\varphi)V^{n+1}\Delta\overline{Q}^{n} - \varphi V^{n-1}\Delta\overline{Q}^{n-1}}{\Delta t} + \overline{Q}^{n}\left[\frac{(1+\varphi)\Delta V^{n} - \varphi\Delta V^{n-1}}{\Delta t}\right] + R^{n+1}$$

Multiple Newton iterations are used to rid the solution of time linearization error at a given time step. A bidirectional Gauss–Seidel solution algorithm is used to solve the resulting linear system.

For quick turnaround time in a design environment, it is essential to parallelize the flow solution algorithm. The present parallel unstructured viscous flow solver is based on coarse-grained domain decomposition for concurrent solution within subdomains and each is assigned to a unique processor. The solver employs MPI message passing for inter-processor communication.

4. SLIDING INTERFACE METHOD

The objective of this work is to develop a sliding interface method [28, 29] that is fast, efficient and addresses some of the previously described limitations for simulations involving relative grid motion. To achieve these objectives, an approach that involves no grid deformation, remeshing, or hole cutting is used. The rotational motion is accomplished by rigidly rotating the subdomain representing the moving component, which is similar to the UVI and structured patch and overlaid methods. The sliding interface method does not impose any restrictions on the subdomain interface.

Since the grid at the interface is arbitrary, the grids on either side of the interface do not align and thus the subdomains are discontinuous. There are two main issues that must be addressed in order to compute the flow across the disjoint subdomain interface: (1) how to compute a flux across the subdomain interface and (2) how to couple the subdomains in the solution process.

4.1. Sliding interface construction

Each subdomain boundary interface is extruded to form part of the sliding interface. Both subdomain interfaces are extruded and overlap in the region near the interface. In 3-D, triangular faces are extruded into prisms and quadrilateral faces into hexahedral elements. This is shown conceptually for a 2-D interface in Figure 1 where the edges on the interface are extruded into quadrilateral elements. For purely notational reasons, one subdomain interface is denoted the primary interface and the other the secondary interface. The one with the finer grid resolution is typically denoted the primary interface. The elements and nodes that are created by the extrusion are denoted as ‘sliding’ elements and ‘sliding’ nodes, respectively. These sliding elements close the control volumes for the nodes on the interface and behave just as interior control volumes. With the control volume closed, a flux can be computed.

The extrusion distance is based on a local measure of the element sizes of the domain being extruding into. A distance metric for each node on the interface is computed as

$$d = \frac{\beta}{n}\sum_{i\in F}\sqrt{(\mathbf{x}_{\text{cell}} - \mathbf{x}_{\text{face}})^2}$$

where F is the set of all faces attached to the node, $\mathbf{x}_{\text{face}}$ is the face centroid attached to the node, $\mathbf{x}_{\text{cell}}$ is the cell centroid of the cell directly atop face i, n is the number of faces attached to the node, and β is a multiplier. A local distance metric is used because the conservation error is of the

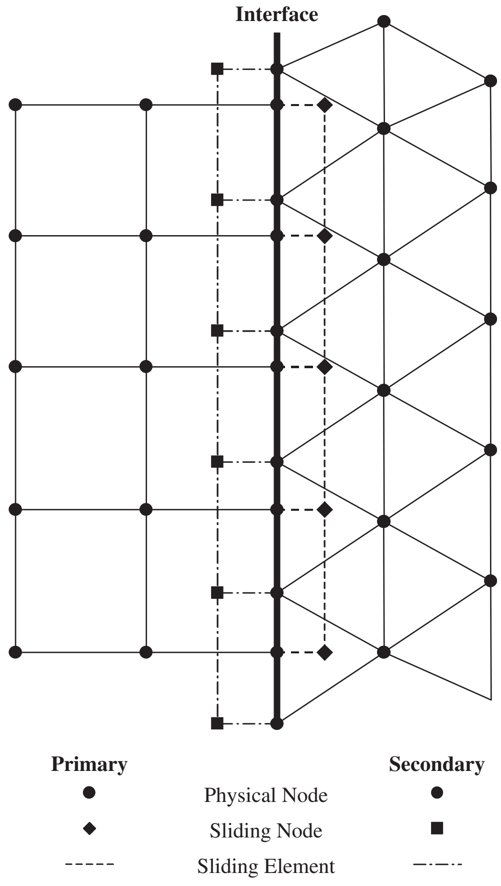

Figure 1. Two-dimensional overlapping sliding interface.

same order of the base conservative scheme if the length of the overlapped region is proportional to the grid spacing [9]. However, this metric provides a local measure of the cell size for the domain being extruded from. In order to keep the information as localized as possible, information about the cell size of the neighbouring subdomain is needed. This is accomplished by an information exchange. For example, for a node on the primary interface, the distance metric information is exchanged with the closest node on the secondary interface, and *vice versa*. After the information exchange, the actual extrusion distance for a node on the primary interface is one half the local distance metric of the closest node on the secondary surface and *vice versa.*

Since a flux across the interface needs to be computed, it is important to obtain the information to compute that flux from nodes as close as possible to the interface, and thus the need for the information exchange to get the extrusion distance based on the domain being extruded into. Note that in general when computing a flux for a face, the information used in the evaluation is local and obtained from surrounding nodes and thus the reasoning for keeping the extrusion close to the interface. For example, if the element size in one subdomain were significantly larger than the neighbouring subdomain, then simply using the distance metric computed for the node would lead to an extrusion distance that might span multiple elements in the neighbouring subdomain. In addition to obtaining information farther away from the interface, this would also allow for the possibility of an extruded element lying inside a solid boundary if one were near the interface. In the vast majority of cases, the procedure described above leads to the extruded elements lying within the elements immediately adjacent to the interface in the neighbouring subdomain.

The closest node information serves as the basis for coupling the interfaces together and provides the locality information to couple the disjoint subdomains. The closest node information is found using a recursive-box algorithm and it has been implemented to work in a parallel computing environment since the closest node may reside in another block. For every node on the primary interface surface, the nearest node on the secondary surface is found and *vice versa*.

For nodes in the isotropic regions of the grid, the extrusion direction is based on the local node normal. However, special consideration is given when extruding into boundary layers. If the local node normal was used and the interface surface did not align with the high aspect ratio boundary-layer grid as shown in Figure 2(a), then the sliding element would span multiple elements in the neighbouring subdomain. This would again lead to using inconsistent or non-local information when computing the flux across the control volume boundaries. Instead, the extrusion direction is computed so it will align with the boundary-layer grid of the neighbouring subdomain, as shown in Figure 2(b), to ensure the information used to compute the flux will be as local as possible.

The extrusion direction is taken to be tangential to the boundary layer elements of the domain being extruded into. For a point on the primary surface, a surface search is performed to find the

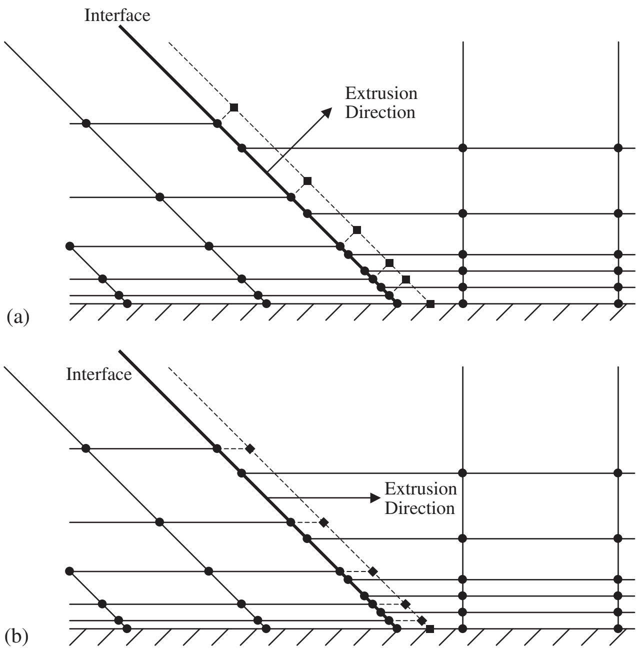

Figure 2. Boundary-layer extrusion direction based on: (a) the local node normal; and (b) adjacent subdomain boundary-layer element information.

containing face on the secondary surface and *vice versa*. The extrusion direction is the direction from the centroid of the containing face to the centroid of the element sitting directly atop the containing face. Another available option for the extrusion direction is that it be parallel to the extrusion direction of the closest boundary node. For a given interface surface, the nearest boundary node is found using the recursive-box algorithm for nodes in the boundary layer. The first option is used for the results presented in Section 6.

Where the interface intersects boundary surfaces, such as a far-field boundary or a solid wall, the interface edges along the boundary are extruded into quadrilateral faces. These extruded faces will then have the same boundary condition applied of the face that it overlaps in the neighbouring subdomain in order to make the appropriate boundary contribution to the flux evaluation for the node on the interface. A fast ray–triangle intersection algorithm is used to determine which boundary face the node should be projected in order to ensure the extrusion direction of the boundary edges are tangent to the neighbouring subdomain boundary [30].

The sliding interface may be constructed every time step using the previously described procedure or, to minimize computational costs, created only once during a preprocessing step. In which case, the extruded interface is rotated into place each time the grid is moved and there is no updating of the local extrusion distance or extrusion direction vector. This is appropriate if the grid spacing on the interface is nearly uniform in the circumferential direction. For the results of the large scale application presented in Section 6, the sliding interface is reconstructed every time step. For the model validation problems present in Section 5, there is no relative grid motion and the interface were constructed only once at problem setup.

4.2. Search algorithm and interpolation

The key for this approach to work efficiently in a parallel computing environment is the search algorithm. A search is performed to find the containing or host element of the sliding nodes in order to interpolate the needed quantities for the sliding node. For grids in relative motion to one another, the grids at the interface are changing every time step and thus the sliding nodes host element changes and a search is required. Note that a search is required for every extruded point, or sliding node, on the interface. The parallel search algorithm uses multiple passes of a volume coordinate search algorithm (or area coordinate search if a point lies on a surface) until all points are found. The basic strategy of the volume search algorithm (and corresponding surface search in 2-D) is based on the neighbouring element algorithm [31]. During the course of the search for a point, should the search algorithm attempt to move across a (parallel) block boundary, the point is transmitted to the appropriate block and the search is continued. To speed the volume search, a good starting location for each point is chosen so that the host element is found in one or two steps. The starting location is taken to be one of the elements attached to the closest node. Using this procedure, the majority of the points are found on the first pass, and only a few (the ones that are near or on block boundaries) require a second pass.

The basis functions for tetrahedral elements can be conveniently expressed using volume coordinates; and note that for a tetrahedral element, the number of nodes and faces are equal. Volume coordinates are the volume ratios of the sub-tetrahedra formed by connecting the nodes of each face to the search point. For non-simplical elements, the element is subdivided into tetrahedra to compute the face-based basis functions needed for the search.

Except for the coordinates of the sliding nodes, all quantities are interpolated. The interpolation is done using the nodal values of the host element and weighting functions. For tetrahedral elements, finite element isoparametric shape functions are used. Given the containing element and coordinates of the sliding grid point, the shape functions can be easily computed. However for the non-simplical elements, computing the shape functions requires the solution of a cubic polynomial and can be computationally expensive. Instead for the pyramid, prism, and hexahedral elements, inverse distance weighting functions are used. The value of the desired quantity for the sliding interface node is interpolated using the weighting functions and the values of the quantity at the nodes of the containing element. With the weighting functions computed, the value of the desired quantity for the sliding interface node is interpolated via

$$\hat{u} = \sum_{j=1}^{n} \phi_j u_j$$

where the $\hat{u}$ is the interpolated quantity, ϕ_j are the finite element shape functions or weighting functions for the containing element, u_j is the value of the quantity at the nodes of the host element, and the summation is over the nodes of the host element. Note that the weighting functions themselves sum to unity. The interpolation involves all the nodes of the host element and as such there is no regard to upwinding or the direction of information travel or wave propagation. This is taken into account during the flux calculation for the node on the interface surface.

4.3. Flux computation

With the values of the vector of conserved variables, $\mathbf{Q}$, known at the extruded sliding interface node, the control volume for the node on the interface can now be closed and a flux computed. Regarding the flux computation, there are several approaches that could be taken when devising the sliding interface. One approach is to conserve the local flux at every point across the interface. Another approach is to conserve the total or global flux across the entire interface. Yet another approach is to compute a flux across the interface without regard to conservation. For this work, the flux is not explicitly conserved across the interface. Note that in a discrete sense, conservation means that all interior fluxes sum to zero and that the only remaining fluxes are the ones (spatial and temporal) in and out of the discrete domain. The interpolation is done for both subdomain interfaces and a flux is computed across each interface independently of the other and thus the fluxes do not cancel at the subdomain interface and gives rise to the non-conservative aspect of the approach.

Note that local flux conservation guarantees global conservation, but global conservation does not guarantee local conservation. If there is a discontinuity in the flow field, then only satisfying global conservation may not be of much use since the discontinuity may not be preserved, and so the globally conservative and the non-conservative approaches could lead to similar results. There is no way to impose local conservation across an interface without a fully connected interface and the only choices are global conservation or none at all. An exhaustive set of numerical simulations, including steady and unsteady, viscous and inviscid cases, have been performed to examine this issue of the non-conservativeness of the interface flux and for all the cases examined thus far, the present approach has performed well [29]. Using this approach for the flux computation, excellent agreement has been obtained with both theoretical and experimental results. As pointed in Reference [6], the issue with subdomain connectivity is not necessarily one of conservation, but one of grid resolution.

Regardless of whether the flux across the interface is strictly conserved or not, a flux at the interface must be computed, and the problem remains how to compute this flux. Different options exist for computing this flux. One is that the flux itself could be directly interpolated. Another is to interpolate for the values of **Q**, the solution or conserved variable vector, and use these interpolated values to compute the flux. The latter approach is used here. With the flux computed, the residual can now be computed in order to compute a new value of **Q** for the next time step.

The next issue to address is how to couple the disjoint subdomains during the solution process. During the solution process, **Q** and $\Delta\mathbf{Q}$ are not computed explicitly for the sliding nodes. Instead, these values are interpolated. As discussed previously, the approach proposed here is to extrude both subdomain interfaces and form overlapping elements. The overlapping interface allows **Q** for the physical nodes on one side of the interface to use information from the other side of the interface and couples the two subdomains. The degree of coupling depends on the interpolation frequency of **Q** and $\Delta\mathbf{Q}$. The interpolation for $\Delta\mathbf{Q}$ is done at the end of each linear sub-iteration, colour change, and directional sweep and the interpolation for **Q** is done at the end of each Newton iteration to ensure the tightest coupling possible. By updating the interfaces during the solution procedure in the manner described above, the values at the interfaces will be at the most current time step and the interface values are fully implicit and the solution procedure remains fully implicit. Updating in this manner avoids the temporal errors and numerical stability issues that can arise due to explicit updating of the interface boundaries after the solution has been advanced to the next time step, in which case the interface values may be lagged by the previous time step [6].

5. MODEL VALIDATION PROBLEMS

Multiple test cases have been studied to examine the effect of the sliding interface on the flow solution in References [28, 29]. The test cases were designed to determine if discontinuities can be maintained across the extruded interface and examine conservation across the interface. In this paper, two test cases are presented: inviscid, supersonic flow through a diverging nozzle and an inviscid shock tube. The first case was designed specifically to check the mass conservation. The inlet Mach number is 1.0 and the inlet radius is 1.0 and geometry of the nozzle is such that the area ratio of the inlet and exit result in an exit Mach number of approximately 2.0. The configuration is shown in Figure 3. At the mid-span location, there is a spherical interface surface of radius 0.75. The radius of the nozzle at the mid-span location is 1.15 and approximately 40–50% of the flow passes through the interface.

Nine cases were run; the baseline case and eight cases employing the sliding interface, and the results are shown in Table I. The mass imbalance listed in Table I is simply the difference in mass flow between the inlet and outlet surfaces. The per cent difference listed in Table I for the sliding interface solutions are with respect to the baseline solution, %Difference = |(Baseline − Sliding)/Baseline|. The baseline case was run assuming steady flow for 1000 iterations beginning from

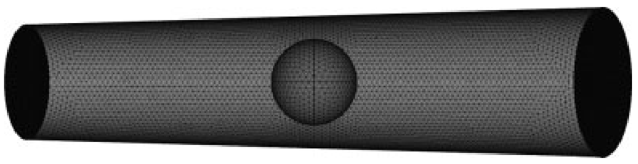
Figure 3. Supersonic, diverging nozzle configuration.

Table I. Comparison of the mass imbalance for the diverging supersonic duct.

Solution	Spherical subdomain	Distance metric multiplier	Mass imbalance	Difference (%)
Baseline	Stationary	Fully connected	5.78797e−5	—
Case1	Stationary	1.00	5.78000e−5	0.138
Case2	Stationary	0.50	5.78489e−5	0.053
Case3	Stationary	0.10	5.78748e−5	0.009
Case4	Stationary	0.01	5.78793e−5	0.001
Case5	Spinning	1.00	5.64444e−5	2.48
Case6	Spinning	0.50	5.65327e−5	2.47
Case7	Spinning	0.10	5.65989e−5	2.21
Case8	Spinning	0.01	5.66135e−5	2.19

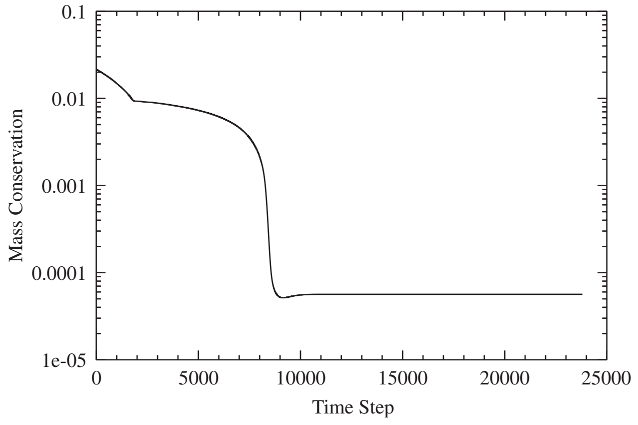

Figure 4. Mass conservation for the rotating sliding interface case.

uniform conditions until the forces and residual remained constant. The first set of sliding interface cases were run using the same conditions and spherical subdomain was kept stationary. Even though the spherical subdomain is not rotating, the flow must still pass through the non-matching interface. During construction of the first sliding grid, the spherical subdomain was rotated 45° about the x-axis to ensure that the primary and secondary sliding interfaces would not align.

Three cases were used to examine the argument for keeping the extrusion close to the interface so that the information needed to compute the flux comes from locations as close as possible to the interface. The distance metric was decreased by a factor of 0.5, 0.1, and 0.01, respectively. It is seen that as the extrusion distance is decreased, the mass imbalance using the sliding interface approaches that of the fully connected baseline case. Thus keeping the extruded or overlapping elements as close as possible to the interface can minimize the conservation errors. This is similar to the results reported in References [6, 9] that demonstrated the interpolation errors at the subdomain interfaces are of the same order as the truncation error of the base conservative scheme. In the limit as the extrusion distance tends to zero, the solutions on each subdomain converge to corresponding solutions of the baseline or fully connected solution at a second-order rate.

For the last set of sliding interface cases, the spherical subdomain was rotated 1° per time step about the x-axis (the axial direction). Due to the grid movement, the simulations were run

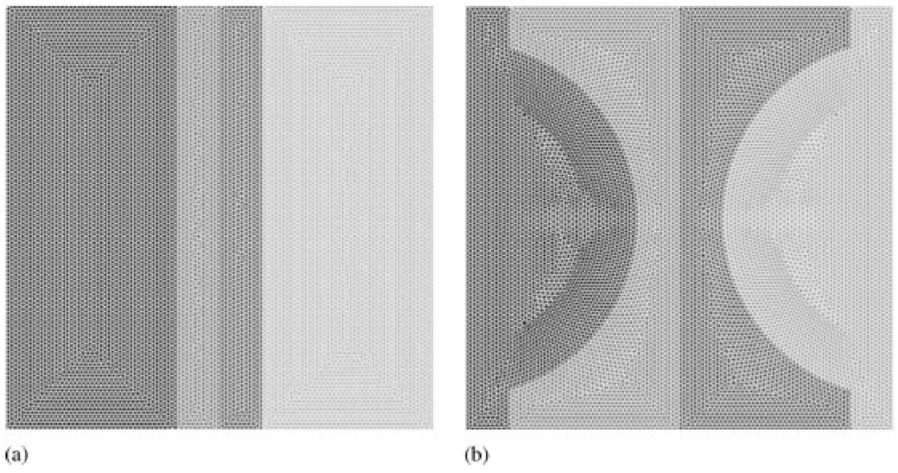

Figure 5. Shock tube grids for: (a) the planar interface; and (b) the curved interface.

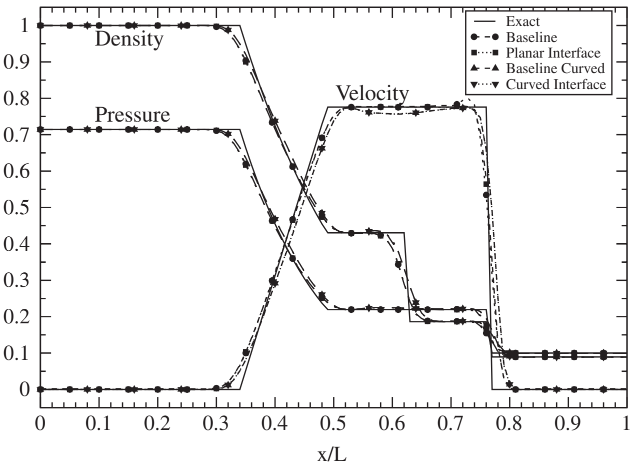

Figure 6. Comparison of pressure, density, and velocity for the theoretical and computed shock tube solutions.

unsteady for nearly 25 000 iterations. Rotating cases were also run where the distance metric was decreased, and the similar trends to the static results were observed as indicated in Table I. Steady state conditions were achieved after approximately 9000 iterations, however the simulation was continued to examine if the rotating sliding interface had any effect on the mass conservation. A plot of the mass conservation time history is shown in Figure 4 for the case using the default distance metric multiplier (1.0). A comparison of the components of the momentum flux and energy flux were also made for each case and revealed percentage differences on the order of 0.01 and smaller.

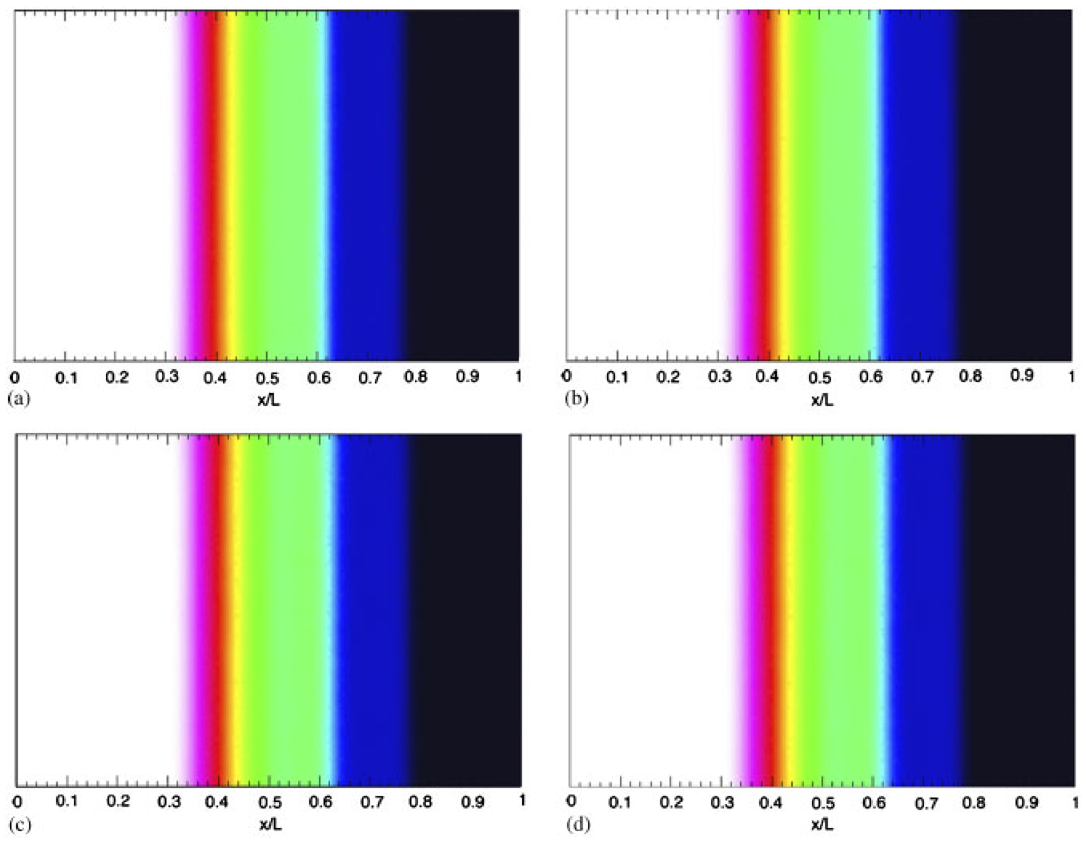

Figure 7. Density contours for: (a) the planar baseline; (b) the planar interface; (c) the curved baseline; and (d) the curved interface solutions.

The second test case is an inviscid shock tube. There is no relative motion in this case, but it was designed to see how unsteady waves pass through the interface. The pressure ratio of the high-pressure gas (on the left of the diaphragm located at $x/L = 0.5$) to the low-pressure gas (on the right of the diaphragm) was 8:1 and the density ratio is 10:1. The extent of the domain is $L \times L \times 0.5L$ and all surfaces of the shock tube are assumed to be inviscid surfaces.

Four grids were constructed, two having planar interfaces and two having curved interfaces. For the first pair, there are two planar interfaces surfaces, one at $x/L = 0.4$ and the other at $x/L = 0.6$, as illustrated in Figure 5(a). A second pair of grids was constructed having curved interfaces as shown in Figure 5(b). For each pair, a baseline grid and a sliding interface grid were constructed. In the baseline grid, the points on the interface surfaces were merged to connect the subdomains and in the sliding interface grid, the points on the interface surfaces were not merged and the subdomains are disjoint.

The solution for each case was run for 160 time steps using a time step of $\Delta t = 0.001$. At $t = 0.16$, the shock wave and contact surface have passed through the interface located at $x/L = 0.6$ and the interface at $x/L = 0.4$ is in the middle of the expansion wave. For the first pair of grids having planar interfaces, the entire wave passes though the interface at the same time. To determine if the shape of the interface affects the waves as they pass through the interface, the second pair of grids with curved interfaces were used. For the curved interface grids, different sections of the waves will be pass through the interface at different times.

A comparison of the computed three-dimensional solutions to the exact one-dimensional solution [32] is shown in Figure 6. The computed solution was taken along $(x, 0.5L, 0.25L)$. Note that even the baseline solution tends to smear the discontinuities. This can be resolved by increasing the number of points in the flow direction. The key idea to note is that all five computed solutions are nearly identical. The various waves were able to pass through all the different interfaces relatively undisturbed.

Shaded density contours for each grid are shown in Figure 7. Overall, the solutions for the sliding interfaces match the baseline results very well. The curved sliding interface exhibits the same characteristics as the curved interface having the grid points merged on the interface. Both slightly shift the location of the shock and contact waves as evident in Figure 7. The mass flow across the interface at $x/L = 0.4$ and at $x/L = 0.6$ was also computed for the shock tube grids. The maximum error for the sliding interface grids compared to the merged baseline grids was less than 0.5%.

6. LARGE SCALE APPLICATION

The missile configuration used for the application case is shown in Figure 8. The missile has a tangent ogive nose, cylindrical fuselage, four moveable canards, and four tail fins. The tail fins are attached to a bearing that spins relative to the fuselage section fore and aft of it. The fuselage has a fineness ratio of approximately 15. Yaw and pitch control is achieved using the canards. The canards are equally spaced around the fuselage in an X-configuration relative to the plus (+)-configuration of the tail fins. For the results presented here, the canards are deflected 16° and held fixed in that orientation as shown in Figure 8. The four tail fins are located at 90-degree increments circumferentially around the tail section and are shown in Figure 8 in a zero degree orientation. The missile geometry also includes various geometrically complex features along the missile fuselage. In addition, there are two railings (located at 0° and 180°) that run nearly the length of the fuselage.

The original motivation behind this simulation was to predict the aerodynamic performance of the missile. When the canards are deflected, the resulting aerodynamic loading imparts a rolling moment on the tail fins causing them to spin, even under steady-state flight conditions. To accurately compute the aerodynamic performance of the missile requires that the tail dynamics be adequately resolved. The spin-rate of the tail is a function of the canard deflection and the free-stream angle of attack. In addition, at low angles of attack, the spin rate is also dependent upon the strength and position of the vortices generated by the canards that convect downstream and impinge on the tail fins. Previous results [33] describing the required grid resolution for accurate prediction of the aerodynamic forces acting on the body and the relative importance of the viscous effects and tail spin rate. Those results were generated using the UVI method and were found to be in good agreement with other grid topology approaches that have been successfully applied to this same problem: Murman *et al.* [34] applied a Cartesian method and both Hall [35] and Nygaard *et al.* [36] used Chimera overset structured grid methods. For the purpose of this work, the results generated using the UVI methods will serve to validate those using the new sliding interface method and

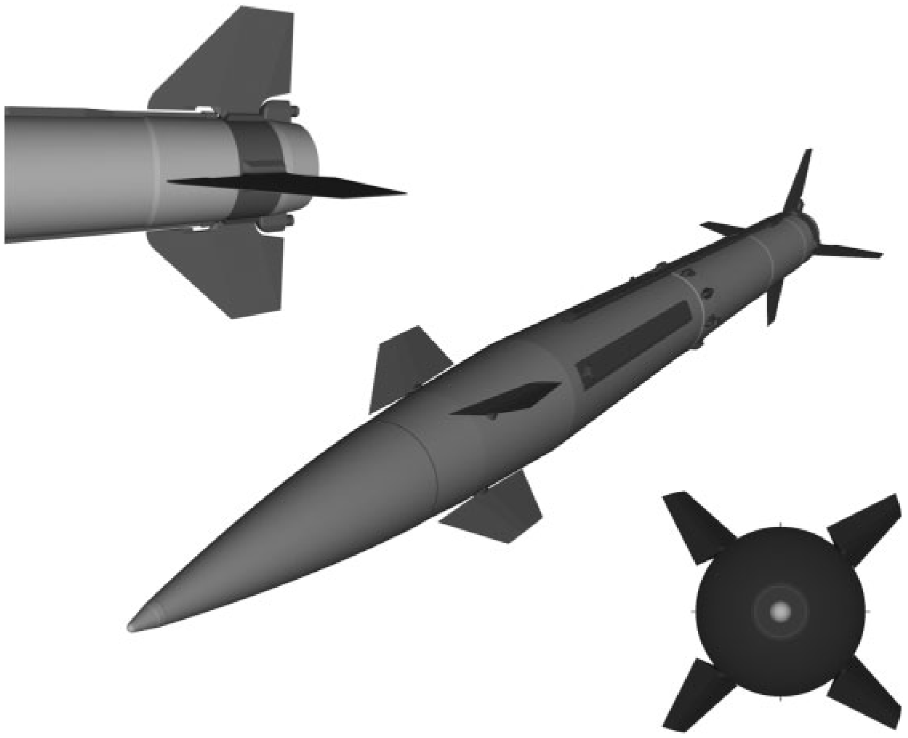
Figure 8. Missile configuration with free-spinning tail.

comparison of these results will be made to the other topology approaches when available. The effect of the sliding interface will be evident since the canard vortices, which have a significant impact on the aerodynamic performance of the missile, must pass through the sliding interface in order to impinge on the tail.

6.1. *Grid construction*

Body-fitted mixed-element type unstructured grids suitable for viscous simulations were generated using AFLR [37]. Both the UVI and sliding interface volume grids were constructed using essentially the same surface grids. An initial normal spacing of 5.0×10^{-5} was used for all the volume grids, which led to $y+$ values of less than 1.0 for the first point from the body indicating good viscous sub-layer resolution.

Significant modifications to the geometry were required for the UVI method. Figure 9 shows a detailed section near the tail. In order for the UVI surface to encompass the tail, the fuselage requires two cuts for the UVI surface to pass through: one before the bearing and one immediately after. As shown in Figure 9(a), both the railings extend underneath the tail fins and there is a very small gap between the fins and the railings to allow the fins to pass over the railings as they spin. To adequately resolve this small gap requires fine point spacing. However due to the memory constraints of a single processor, the required resolution exceeds that allowed for the number of points on the UVI surface. Recall that the UVI method is based on an inherently sequential grid

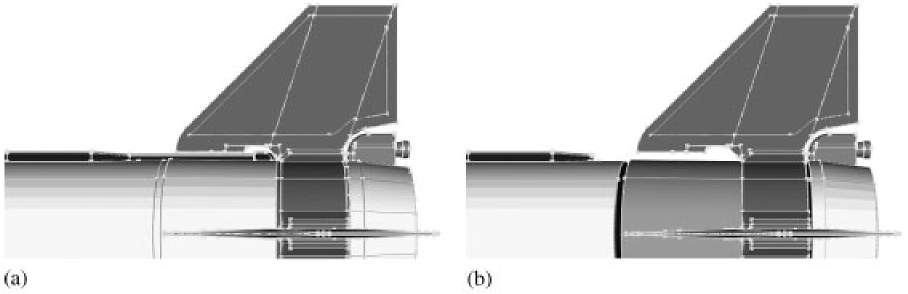

Figure 9. (a) Original geometry near the tail section; and (b) geometric modifications to the railings and fuselage for the UVI method.

generation reconnection process, and as such, the entire UVI surface and all volume elements connecting to the surface must reside on a single processor. Therefore both railings were truncated to avoid resolving the gaps and modified railings are shown in Figure 9(b). Note that part of the stationary fuselage is now within the UVI surface and will be rotated. To counteract the effect of this spinning surface, a rotating viscous boundary condition was applied to this surface with a spin rate equal and opposite to that of the tail. Where the fuselage was cut to allow the UVI surface to pass through, a blockage was defined to turn off portions of the field grid using a tapered cylinder to represent the missing portion of the fuselage. Edges with both nodes inside the region definition were turned off and edges with only one node inside the region were treated as viscous edges. The UVI volume grid contains approximately 9.5M grid points.

Two grids for use with the sliding interface method were constructed. In order to make a direct comparison with the UVI method, the first sliding interface grid is the same as the UVI grid except that the nodes on the interface surface surrounding the tail are not merged. The second sliding interface grid is constructed with no modifications to the fuselage or railing. The same volume grid generation parameters were used as for the UVI grid so that the field grids would be as similar as possible. However, extending the railings and resolving the gap between the fins and the railings adds approximately two million nodes to the Sliding2 grid for a total of slightly more than 12M grid points.

6.2. Simulation results

The simulations were performed using the following parameters: $M = 1.6$, $\alpha = 4°$, $\omega = 0$ and 2500 rpm, and $Re_L = 7.01 \times 10^6$ based on the missile length. This angle of attack was selected since it was expected to have the strongest interaction between the canard vortices and the tail. The force and moment coefficients are defined using the standard normalization by dynamic pressure. The reference area S was taken to be the cross-sectional area of the fuselage and the reference length c was taken as the fuselage diameter. The moments were taken about a point at $x/L = 0.66$ located on the centreline. Preliminary simulations were performed to determine the proper solver parameters to ensure the unsteady solution was converged at each time step. Solutions were obtained using time steps corresponding to 0.1°, 0.5°, 1.0°, and 2.0° of rotation per time step, and the nature of the problem dictated a time step no larger than 0.5° of rotation was necessary.

The canards generate strong horseshoe vortices that convect the length of the missile. At this low angle of attack, the vortices are also close enough to the missile body to effect the force distribution on the fuselage and tail. With the asymmetrical canard deflection used here, the two canards on the upper surface generate tip vortices rotating in a negative sense about the longitudinal axis (or counter-clockwise if viewed from the nose looking aft). The corresponding canard root vortices are rotating in a positive sense (clockwise) about the longitudinal axis. The lower two canards generate clockwise tip vortices and counterclockwise root vortices. The strongest vortex is generated by the upper port canard since it is deflected leading edge upwards and sees the largest relative angle of attack. Due to the angle of attack and deflection of the canards, as the vortices travel downstream they convect upwards and to the port side. By the tail section of the missile, 3 of the 4 pairs of vortices are located in the upper port quadrant. The fourth pair, the one generated by the lower starboard canard, impinges on the fuselage. The root vortices from the upper and lower port canards appear to be wrapped around the stronger tip vortices as they convect downstream.

The above described vortex system leads to an asymmetrical inflow condition for the tail fins. The asymmetry of the flow results in an asymmetrical pressure distribution on the tail fins. The canards, which cause a starboard motion of the missile, combined with this vortex induced pressure difference on the fins produce a positive rolling moment about the longitudinal axis and cause the tail to spin. The direction to spin the tail was determined from the zero spin-rate case, i.e. the tail is held fixed.

The static results for the UVI and sliding interface solutions are shown in Table II and the roll-averaged results for the $\omega = 2500$ rpm tailspin rate are shown in Table III. There is very little difference between the UVI and Sliding1 results. However, there is an appreciable difference between the Sliding1 and Sliding2 results. This is due to the difference in geometry between the two configurations, namely the additional segment of railing in the Sliding2 grid. Also included in Table II are the results generated using the OVERFLOW-D flow solver [36]. The axial force coefficient for the OVERFLOW-D results is omitted because it did not include the base drag. Except for the pitching moment coefficient, there is excellent agreement with the results predicted using OVERFLOW-D.

The time history for the rolling moment coefficient is shown in Figure 10 and the Sliding1 and UVI results are again very similar. The differences in the Sliding2 results are due to the presence

Table II. Roll-averaged force and moment coefficients for the static case ($\omega = 0$ rpm).

Solution	C_A	C_Y	C_N	C_l	C_m	C_n
UVI	0.787	0.607	1.083	0.453	0.297	−6.317
Sliding1	0.788	0.607	1.085	0.454	0.289	−6.319
Sliding2	0.808	0.602	1.097	0.455	0.246	−6.344
OVERFLOW		0.607	1.06	0.483	0.439	−6.44

Table III. Roll-averaged force and moment coefficients for $\omega = 2500$ rpm.

Solution	C_A	C_Y	C_N	C_l	C_m	C_n
UVI	0.794	0.585	1.152	−0.119	−2.00e−2	−6.415
Sliding1	0.795	0.585	1.153	−0.119	−2.52e−2	−6.418
Sliding2	0.811	0.580	1.169	−0.119	−8.40e−2	−6.446

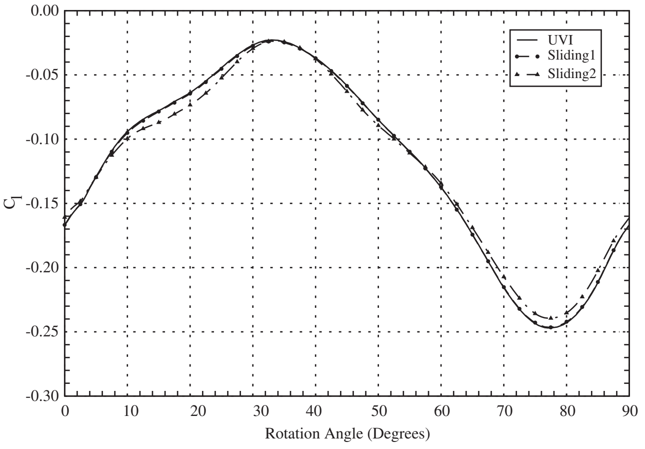

Figure 10. Rolling moment coefficient for the $\omega = 2500$ rpm tail spin rate.

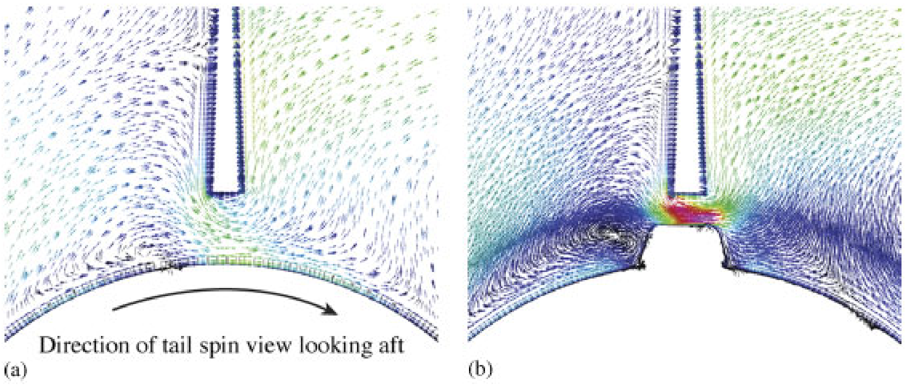

Figure 11. Crossflow velocity vectors near the tail for: (a) the UVI solution; and (b) the Sliding2 solution at the $\omega = 2500$ rpm tail spin rate.

of the railing as the tail fin passes overhead. A comparison of the cross-flow velocity vectors taken on a cutting plane in the vicinity of where the tail passes over the top railing is shown in Figure 11 for the $\omega = 2500$ rpm tail spin rate solution. Note the railing has been truncated and is no longer present in the UVI and Sliding1 solutions. The presence of the railing causes the flow in the gap between the railing and the fin to accelerate to nearly 45% of the free-stream value. The flow in this region is nearly twice as fast as compared to the flow in the same region (without the railing) in the UVI and Sliding1 solutions. Also note the UVI and Sliding1 solutions are nearly identical.

As discussed previously, the vortex interaction has a significant effect on the missile performance. The fact that the sliding interface results are similar to the UVI results is important since the vortices must pass through the sliding interface to interact with the tail. Thus, it appears the vortices are passing through the sliding interface without any change in strength or position and the interface is not having any noticeable impact on the results. As discussed before, there is very little difference in the Sliding1 and UVI solutions indicating little to no influence by the sliding interface. The differences in the Sliding2 results are due to geometric differences in the grids.

A comparison of the helicity contours near the tail for each solution is shown in Figure 12. The planes on which helicity is plotted are $x/L = 0.96$ and 1.05. The contours are very similar and thus the vortices pass through the sliding interface without any additional dissipation or distortion.

6.3. *Computational expense*

Each solution utilized 64 processors on a Linux super-cluster. The cluster is comprised of 192 IBM× 335 nodes and uses 100 Mb/s Ethernet switches for inter-node communication. Individual nodes contain dual 3.06 GHz Xeon processors and 2.5 GB of RAM. The run time information is listed in Table IV and all times are listed in seconds. Note the Sliding2 grid is significantly larger than the other two grids yet had a shorter run time than did the UVI case. In each case, a quarter revolution can be run in less than 9 hours and the Sliding1 case in under 5 hours. The parallel efficiency measures are listed in Table IV. To get an idea of the percentage of time the remaining processors are idle, the following estimate is used:

$$\%\mathrm{Idle} = \frac{T_{\mathrm{reconn}}(\mathrm{np} - \mathrm{nuvi})}{T_{\mathrm{Total}}} \cdot 100$$

where T_{reconn} is the total time spent in the rotation and reconnection process for all UVI processors, np is the total number of processors, nuvi is the number of UVI processors, and T_{Total} is the sum of the run times for the individual processors. Note the for the sliding interface cases, nuvi is the

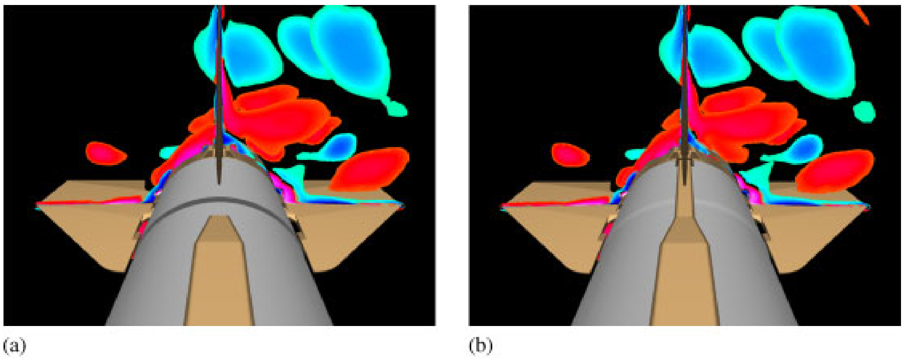

Figure 12. Helicity contours near the tail for: (a) the UVI solution; and (b) the Sliding2 solution at the $\omega = 2500$ rpm tail spin rate.

Table IV. Computational expense for the free-spinning tail missile simulations.

Grid	Pts. per processor	CPU time per time step	Wall time per time step	Wall time per 1/4 revolution
UVI	148 000	59.32	175.16	31 528
Sliding1	148 500	56.31	96.13	17 304
Sliding2	188 074	82.78	164.30	29 574

Table V. Parallel efficiency measures for the free-spinning tail missile simulations.

Grid	Total run time	Total communication time	Total rotation/reconnection time	nuvi	Idle (%)
UVI	2 024 804	547 898	9032	1	28.10
Sliding1	1 114 747	38 319	116	22	0.44
Sliding2	1 902 372	46 039	177	32	0.30

number of processors or partitions having sliding interface surfaces. The results are listed in Table V and all times are listed in seconds. With just a single UVI processor, the UVI solution incurs significantly more idle time and a correspondingly longer run time. Thus the UVI method has a longer run time because the majority of the processors, 63 of 64, are idle during the reconnection process.

7. SUMMARY AND CONCLUSIONS

A sliding interface method has been developed for simulations involving relative rotational grid motion. The method alleviates computationally expensive grid deformation, remeshing, and hole cutting procedures. Rotational relative motion is accomplished by rigidly rotating a subdomain representing the moving component. At the subdomain interface boundary, the faces along the interfaces are extruded into the adjacent subdomain to create new volume elements and provide a one-cell overlap. These new volume elements close the control volumes for the nodes on the interface surface and allow a flux to be computed across the subdomain interface. An interface flux is computed independently for each subdomain, and in doing so the method is not strictly conservative across the subdomain interface. The values of the solution variables and other quantities for the nodes created by the extrusion process are found by interpolation. The extrusion is done so that the interpolation will maintain information as localized as possible. A parallel implementation of the neighbour search is used to find the extruded points in the adjacent subdomain.

The method has been implemented in a parallel, node-centred finite volume, high-resolution viscous flow solver. The method developed is efficient and has been shown to be faster than a current state-of-the-art method, the UVI method, for unstructured grid applications. The method does not impose any restrictions on the subdomain interface aside from the requirement that the axisymmetric limitation required for rotational relative motion. The grid on the subdomain interface can be arbitrary. The boundary surfaces between the two subdomains can have completely independent grids from one another; meaning they do not have to connect in a one-to-one manner and no symmetry or pattern restrictions are placed on the surface grid.

To address interface flux conservation and validate the method, numerical simulations were performed for flow through a diverging duct and a shock tube. Fully-connected grids without a sliding interface were constructed for comparison. For the diverging duct test case, the net mass flow rate using the sliding interface was virtually identical to that for the baseline grid. It was demonstrated that minimizing the extrusion distance helps minimize the conservation errors. Rotating the sliding interface had no effect on the mass flow rate either. The shock tube test case showed that the sliding interface had no adverse effect on the unsteady waves that passed through it.

The sliding interface method was also demonstrated and validated on a large-scale geometrically complex case. The free-spinning tail missile case highlighted some of the advantages of the sliding interface method compared to the UVI method. The sliding interface required no geometric modifications and had significantly shorter run times. Comparisons of the roll-averaged and time history data were made to the UVI and OVERFLOW-D numerical solutions and excellent agreement was found. For both the small-scale model problems and the large-scale applications, there are no apparent adverse effects on the numerical solutions by not strictly enforcing flux conservation at the subdomain boundary.

NOMENCLATURE

α	angle of attack
Δt	time step
ω	angular velocity, rate of rotation
C_{A}	axial force coefficient
C_{N}	normal force coefficient
C_Y	side force coefficient
C_l	rolling moment coefficient
C_m	pitching moment coefficient
C_n	yawing moment coefficient
L	reference length
M	Mach number
Re_{L}	Reynolds number
$\mathbf{Q}$	vector of dependent variables

REFERENCES

1. Liu Z, Hill DL. Issues surrounding multiple frames of reference models for turbo compressor applications. *Proceedings of the 15th International Compressor Engineering Conference at Purdue University*, West Lafayette, IN, 2000.
2. Löhner R, Baum JD. Three-dimensional store separation using a finite element solver and adaptive remeshing. *AIAA*: *Aerospace Sciences Meeting*. AIAA: Washington, DC, 1991.
3. Batina JT. Unsteady Euler airfoil solutions using unstructured dynamic meshes. *AIAA Aerospace Sciences Meeting*. AIAA: Washington, DC, 1989.
4. Degand C, Farhat C. A three-dimensional torsional spring analogy method for unstructured dynamic meshes. *Computers and Structures* 2000; **80**:305–316.
5. Steger J, Dougherty FC, Benek JA. A Chimera grid scheme. In *Advances in Grid Generation*, Ghia KN, Ghia U (eds), vol. 5. ASME FED, 1983; 59–69.
6. Meakin RL. On the spatial and temporal accuracy of overset grid methods for moving body problems. *AIAA Applied Aerodynamics Conference*. AIAA: Washington, DC, 1994.
7. Meakin RL, Wissink AM. Unsteady aerodynamic simulations of static and moving bodies using scalable computers. *AIAA Computational Fluid Dynamics Conference*. AIAA: Washington, DC, 1999.
8. Meakin RL. Unsteady simulation of the viscous flow about a V-22 rotor and wing in Hover. *AIAA Atmospheric Flight Mechanics Conference*. AIAA: Washington, DC, 1995.
9. Wang ZJ, Hariharan N, Chen R. Recent developments on the conservation property of Chimera. *AIAA Aerospace Sciences Meeting*. AIAA: Washington, DC, 1998.
10. Cali PM, Couaillier V. Conservative interfacing for overset grids. *AIAA Aerospace Sciences Meeting*. AIAA: Washington, DC, 2000.
11. Nakahashi K, Togashi F. Intergrid-boundary definition method for overset unstructured grid approach. *AIAA Journal* 2000; **38**(11):2077–2084.

12. Löhner R, Sharov D, Luo H, Ramamurti R. Overlapping unstructured grids. *AIAA Aerospace Sciences Meeting*. AIAA: Washington, DC, 2001.
13. Luo H, Sharov D, Baum JD, Lhner R. An overlapping unstructured grid method for viscous flows. *AIAA Computational Fluid Dynamics Conference*. AIAA: Washington, DC, 2001.
14. Togashi F, Ito Y, Nakahashi K, Obayashi S. Overset unstructured grids method for viscous flow computations. *AIAA Applied Aerodynamics Conference*. AIAA: Washington, DC, 2003.
15. Zang SJ, Liu J, Chen Y-S, Zhao X. A dynamic mesh method for unstructured flow solvers. *AIAA Computational Fluid Dynamics Conference*. AIAA: Washington, DC, 2003.
16. Pan D, Chao M-J, Chien SK. Euler computations of rotor flows on unstructured rotating meshes. *Journal of Aircraft* 2001; **38**(4):672–679.
17. Yu W-S, Kunz RF, Ettorre SM, Antal SP. Unstructured rotor-stator analysis of axial turbomachinery using a pressure based method. *Proceedings of International Mechanical Engineering Congress and Exposition*, New York, 2001
18. Mathur SR. Unsteady flow simulations using unstructured sliding meshes. *AIAA Fluid Dynamics Conference*. AIAA: Washington, DC, 1994.
19. Sreenivas K, Hyams DG, Sheng C, Jayaraman B, Wang X, Mitchell B, Jiang MY, Pankajakshan R, Taylor LK, Gaither KP, Gaither A, Beeland H, Marcum DL, Briley WR, Whitfield DL. Physics-based maneuvering simulations for tiltrotor aircraft. *MSSU-COE-ERC-02-06*, Mississippi State University, Starkville, MS, April 2002.
20. Marcum DL, Weatherill NP. Unstructured grid generation using iterative point insertion and local reconnection. *AIAA Journal* 1995; **33**(9):1619–1625.
21. Janus JM, Whitfield DL. Counterrotating prop-fan simulations which feature a relative-motion multiblock grid decomposition enabling arbitrary time steps. *AIAA Aerospace Sciences Meeting*. AIAA: Washington, DC, 1990.
22. Venkatakrishnan V, Mavriplis DJ. Implicit method for the computation of unsteady flows on unstructured grids. *ICASE Report 95-60*, 1995.
23. Hyams DG, Sreenivas K, Sheng C, Briley WR, Marcum DL, Whitfield DL. An investigation of parallel implicit solution algorithms for incompressible flows on multielement unstructured topologies. *AIAA Aerospace Sciences Meeting*. AIAA: Washington, DC, 2000.
24. Roe P. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics* 1981; **43**:357–372.
25. Barth TJ, Jesperson DC. The design and application of upwind schemes on unstructured meshes. *AIAA Aerospace Sciences Meeting*. AIAA: Washington, DC, 1989.
26. Spalart PR, Allmaras SR. A one-equation turbulence model for aerodynamic flows. *AIAA Aerospace Sciences Meeting*. AIAA: Washington, DC, 1992.
27. Whitfield DL, Taylor L. Discretized Newton-relaxation solution of high resolution flux-difference split schemes. *AIAA Fluid Dynamics*, *Plasma Dynamics and Laser Conference*. AIAA: Washington, DC, 1991.
28. Blades EL, Sreenivas K, Hyams DG. Arbitrary overlapping interfaces for unsteady unstructured parallel flow simulations. *AIAA Aerospace Sciences Meeting*. AIAA: Washington, DC, 2003.
29. Blades EL. A sliding interface method for unsteady unstructured parallel flow simulations. *Ph.D. Dissertation*, Mechanical Engineering Department, Mississippi State University, Starkville, MS, 2004.
30. Möller T, Trumbore B. Fast, minimum storage ray/triangle intersection. *Journal of Graphics Tools* 1997; **2**(1): 21–28.
31. Löhner R, Ambrosiano J. A vectorized particle tracer for unstructured grids. *Journal of Computational Physics* 1990; **91**:22–31.
32. Anderson JD. *Modern Compressible Flow with Historical Perspective* (2nd edn). McGraw-Hill: New York, 1990.
33. Blades EL, Marcum DL. Navier–Stokes simulation of a missile with a free-spinning tail using unstructured grids. *AIAA Aerospace Sciences Meeting and Exhibit*. AIAA: Washington, DC, 2004.
34. Murman S, Aftosmis M. Cartesian-grid simulations of a canard-controlled missile with a spinning tail. *AIAA Applied Aerodynamics Conference*. AIAA: Washington, DC, 2003.
35. Hall L. Simulation of a missile with spinning tail fins using Chimera moving body methodology. *AIAA Aerospace Sciences Meeting and Exhibit*. AIAA: Washington, DC, 2004.
36. Nygaard T. Aeromechanic analysis of a missile with freely spinning tailfins. *AIAA Applied Aerodynamics Conference*. AIAA: Washington, DC, 2003.
37. Marcum DL. Efficient generation of high quality unstructured surface and volume grids. *Engineering with Computers* 2001; **17**(3):211–233.